Project Summary: The MetaScience Extreme Modeling Vision

Keywords: Optimization, Modeling, Supercomputing, Evolution, Diversification, AutoDifferentiation

CRI Concept: Transformative build-out of a unified mathematical optimization modeling paradigm infrastructure, MetaCalculus, into a diversified science & engineering infrastructure, MetaScience.

MetaCalculus is the current version of a proven extreme modeling paradigm and solution platform able to produce new R&D applications in sufficiently short times to meet narrow windows of opportunity that often arise in research and product development. This proposal is to transfer MetaCalculus from serial computers to an HPC supercomputer and provide the plan for a pilot university-community build-out of higher diversified modeling techniques built upon this platform.

Our intent is to apply the resources of the Center for Computation Sciences (CCS) at the University of Kentucky (UK) as the CI:ADDO outreach center and laboratory to produce interfaces for interactive optimization supercomputing, renewing a practice of MetaCalculus on early supercomputers on which it was marketed through the 1970s, but now extended to the full power of modern multi-core parallelism. The laboratory resources of CCS, and the UK supercomputer and staff, were chosen for this pilot project primarily due to the special relationship between Professor J. M. McDonough at UK and the team who developed the technology, headed by the PI, J. M. Thames.

This CI-P planning project will first transfer an ensemble of proprietary and open-source software technology to UK/CCS by installing it on the UK supercomputer and making parts of it available on other computers for access by CCS staff, faculty, and students. Then training courses and webinars will be conducted by McDonough, Thames, and other team members, beginning in the spring of 2012, thus providing the initial educational component to this project. This will be followed by recruiting participants from UK, and other universities, to participate in the proposed CI:ADDO follow-on project and its proposal preparation, beginning in late-summer of 2012.

***Intellectual Merit**—This project will continue development of higher-order "extreme modeling" approaches that can achieve rapid prototyping of new concepts and products in the short time frames that students and faculty have to accomplish significant research and that industry has to demonstrate viability of new concepts for commercialization. This involves the automation and evolution of software from the complexity of the base parallel hardware toward higher levels of abstract conceptualization by humans. Unlike other high-level programming languages, MetaCalculus employs an advanced form of automatic differentiation (AD)—including (automatic) chain-rule propagation to any level—as well as a suite of numerical analytic techniques (similar to those in MatLab) within its syntax. This provides the means to use available software technology and university resources to advance the art of modeling and programming to create scientific demand to match the device/hardware output of our semiconductor industry, potentially bringing ease of use of supercomputing to all engineers and scientists. It is expected that solution of advanced multi-disciplinary optimization problems made possible by MetaCalculus will be widely reported at conferences and in the archival literature.*

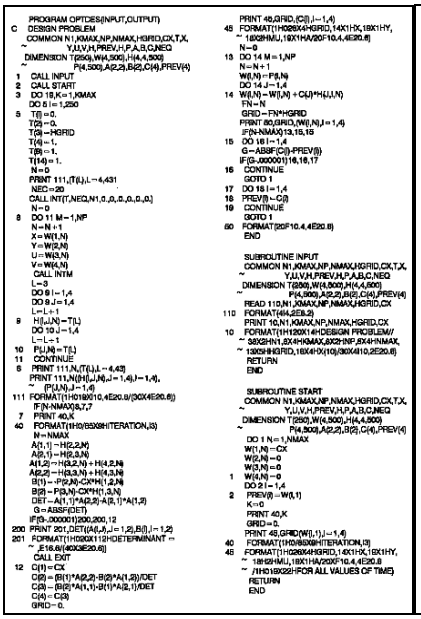
***Broader Impact**—This implementation and further development of MetaCalculus on UK's supercomputer cluster, available to the U.S. scientific community and industry through our proposed broad university system (expected to include some universities in the HBCU catagory), can energize and broaden the phenomenal growth of the open-source software community toward cost-effective one-of-a-kind R&D innovations when these are appropriate, as opposed to little more than recycling staple software like Unix, for example. In particular, wide availability of MetaCalculus will produce initiation of software recycling of a different kind, namely, converting legacy simulation software into optimization software via the "extended" Fortran embodied in MetaCalculus. This is an increasingly important capability in industry and academia as the need to find optimal solutions increases.*

This will allow engineers and scientists to apply the sophisticated optimization tools of MetaCalculus to recycle models and create new sophisticated models now needed in the context of very complex multi-disciplinary problems. Moreover, because code generation is relatively simple, and a wide range of numerical methods is encapsulated within MetaCalculus, the engineer/scientist can master the language very quickly and can then address (code) very sophisticated problems without the need to debug numerical algorithms. Clearly, this will result in significantly increased productivity of engineers and scientists in all disciplines by removing the "division of labor" (between engineer and computer scientist/mathematician) that existed in the past when the various still widely-used legacy codes were developed.

Project Description

Computing Research Infrastructure Concept

MetaCalculus is a paradigm and infrastructure which adds templates for three tiers of holistic modeling: *simulation*, *correlation*, and *optimization* to conventional algebraic programming languages. Together these three tiers comprise a modeling alphabet featuring combination by hidden *algorithm nesting*, like subroutines; thus the use of the word "tier". The following diagrams illustrate the progression of abstraction of the same application from ordinary Fortran on the left, to Fortran Calculus on the right. Figures 2 and 3 represent a 3rd generation and a 7th generation in the evolution of this paradigm as it exists today, respectively.



```
      PROGRAM OPTDES(INPUT,OUTPUT)
C     DESIGN PROBLEM
      COMMON N1,KMAX,NP,NMAX,HGRID,CX,T,X,
     ~        Y,U,V,H,PREV,H,P,A,B,C,NEQ
      DIMENSION T(250),W(4,500),H(4,4,500)
     ~        P(4,500),A(2,2),B(2),C(4),PREV(4)
    1 CALL INPUT
    2 CALL START
    3 DO 18,K=1,KMAX
        DO 5 I=1,250
    5   T(I)=0.
        T(2)=-0.
        T(5)=HGRID
        T(4)=1.
        T(8)=1.
        T(14)=1.
        N=0
        PRINT 111,(T(L),L=4,431
        NEC=20
        CALL INT(T,NEQ,N1,0,.0,.0,.0,.0,)
        N=0
    8   DO 11 M=1,NP
        N=N+1
        X=W(1,N)
        Y=W(2,N)
        U=W(3,N)
        V=W(4,N)
        CALL INTM
        L=3
        DO 9 I=1,4
        DO 9 J=1,4
        L=L+1
    9   H(I,J,N)=T(L)
        DO 10 J=1,4
        L=L+1
   10   P(J,N)=T(L)
   11   CONTINUE
        PRINT 111,N,(T(L),L=4,43)
        PRINT 111,N,((H(I,J,N),J=1,4),I=1,4),
     ~     (P(J,N),J=1,4)
  111 FORMAT(1H0I8/(0I10,4E20.8/(30X4E20.8)))
        IF(N-NMAX)8,7,7
    7   PRINT 40,K
   40   FORMAT(1H0/8X9HITERATION,I3)
        N=NMAX
        A(1,1)=H(2,2,N)
        A(2,1)=H(3,2,N)
        A(1,2)=H(2,3,N)+H(4,2,N)
        A(2,2)=H(3,3,N)+H(4,3,N)
        B(1)=-P(2,N)-CX*H(1,2,N)
        B(2)=-P(3,N)-CX*H(1,3,N)
        DET=A(1,1)*A(2,2)-A(2,1)*A(1,2)
        C=ABS(F(DET)
        IF(5-.000001)200,200,12
  200 PRINT 201,DET((A(I,J),J=1,2),B(I),I=1,2)
  201 FORMAT(1H0023(1PE10E3)ETERMINANT =
     ~  ,E16.8/(40X3E20.8))
        CALL EXIT
   12   C(1)=CX
        C(2)=(B(1)*A(2,2)-B(2)*A(1,2))/DET
        C(3)=(B(2)*A(1,1)-B(1)*A(2,1)/DET
        C(4)=C(3)
        GRID=0.
        PRINT 45,(GRID,(C(I),I=1,4)
   45   FORMAT(1H0026X4HGRID,14X11HX,15X1HY,
     ~     16X8HMU,16X1HA/20F10.4,4E20.8)
        N=0
   15   DO 14 M=1,NP
        N=N+1
   14   W(M,N)=P(M,N)
        DO 14 J=1,4
   14   W(M,N)=W(M,N)+C(J)**H(J,M,N)
        FN=N
        GRID=FN*HGRID
        PRINT 60,(IRC(I),W(I,N),I=1,4)
        IF(N-NMAX)13,15,15
   15   DO 16 I=1,4
        C=-ABS(F(C))-PREV(I))
        IF(C-.000001)16,16,17
   16   CONTINUE
        GOTO 1
   17   DO 18 I=1,4
   18   PREV(I)=C(I)
   19   CONTINUE
        GOTO 1
   60   FORMAT(20F10.4,4E20.8)
        END

        SUBROUTINE INPUT
        COMMON N1,KMAX,NP,NMAX,HGRID,CX,T,X,
     ~        Y,U,V,H,PREV,H,P,A,B,C,NEQ
        DIMENSION T(250),W(4,500),H(4,4,500)
     ~        P(4,500),A(2,2),B(2),C(4),PREV(4)
        READ 110,N1,KMAX,NP,NMAX,HGRID,CX
  110   FORMAT(4I4,2E8.3)
        PRINT 10,N1,KMAX,NP,NMAX,HGRID,CX
   10   FORMAT(1H1/20X14HDESIGN PROBLEM//
     ~  38X2HN1,5X4HKMAX,6X2HNP,5X4HNMAX,
     ~  13X9HHGRID,16X4HX(10)/(30X4I10,2E20.8)
        RETURN
        END

        SUBROUTINE START
        COMMON N1,KMAX,NP,NMAX,HGRID,CX,T,X,
     ~        Y,U,V,H,PREV,H,P,A,B,C,NEQ
        DIMENSION T(250),W(4,500),H(4,4,500)
     ~        P(4,500),A(2,2),B(2),C(4),PREV(4)
        DO 1 N=1,NMAX
        W(1,N)=CX
        W(2,N)=0
        W(3,N)=0
        W(4,N)=0
        DO 2 I=1,4
    2   PREV(I)=W(I,1)
        K=0
        PRINT 40,X
        GRID=0.
        PRINT 45,GRID(W(I,1),I=1,4)
   40   FORMAT(1H0/8X9HITERATION,I3)
   45   FORMAT(1H0026X4HGRID,14X11HX,15X1HY,
     ~  16X8HMU,16X1HA/20F10.4,4E20.8
     ~  /1H0/8X22HFOR ALL VALUES OF TIME)
        RETURN
        END
```

Figure 1.Fortran IV Version

```
/ ***** OPTIMAL DESIGN AND CONTROL EXAMPLE ***** BELLMAN AND KALABA
/       X, STATE VARIABLE
/       Y, CONTROL VARIABLE
/       UI=MU, LAGRANGE MULTIPLIER
/       A    , DESIGN PARAMETER
/
READ DATA
   MINIMIZE J
      INDEPENDENT A
         SOLVE GY
            VARY Y0
            Y = Y0
            X=1
            U=A
            Z=1
            T=0
            INTEGRATION DERIVS,T,,,1,Z,ZDOT,X,XDOT,Y,YDOT,U,UDOT
            DO UNTIL T GE 1
               INTEGRATE DERIVS
            REPEAT
            GY=Y
         END LOOP
      J = Z/2 + A**2/2
      PRINTOUT T,X,Y,U,A,J
   END LOOP
   BLOCK DERIVS
      ZDOT = X**2 + Y**2
      XDOT = -A*X + Y
      YDOT = X + A*Y
      UDOT = Y*X
   END BLOCK
   PRINTOUT J,A
   STOP
   END
```

Figure 2. SLANG Version

```
GLOBAL ALL
PROBLEM OPTDES
~   FIND A; IN TWOPT; BY HERA;
    TO MINIMIZE OBJ
END

MODEL TWOPT ! Two point boundary value problem
  Y0=.7  !  Guess for initial condition
  FIND Y0; IN TRAJ; BY AJAX; TO MATCH Y1
  OBJ=Z/2+A**2/2
END

MODEL TRAJ     ! Initial value problem
  Y=Y0 : X=1 : Z=0 : T=0 ! Initial conditions
  DT=0.25 : TF=1
  INITIATE ISIS; FOR DIFF;
~     EQUATIONS ZDOT/Z, XDOT/X, YDOT/Y;
~     OF T; STEP DT; TO TF
  INTEGRATE DIFF; BY ISIS
  Y1=Y ! Boundary condition
  TERMINATE DIFF
END

MODEL DIFF
  ZDOT=X**2+Y**2 ! Objective differential equation
  XDOT=-A*X+Y    ! State differential equation
  YDOT=X+A*Y     ! Control differential equation
END
```

Figure 3.Fortran Calculus Version

Figure 4 shows the same application in the 6th generation language, TSPROSE, This image is the copy of a figure from the R/D Magazine article *Computing in Calculus* [5], which shows its code (with editor line numbers) imbedded in a hierarchy diagram to illustrate the sub-problem nesting. This was a commercial product marketed in the time-sharing era, and featured interactive optimization. *Today, this paradigm offers the potential to transition the computer-science collaboration known as "extreme programming" into a peer-science collaboration of "extreme modeling".* It is the result of a unique semantic architecture, which implements *nested algorithmic differentiation* (NAD) [16][19][20].

Infrastructure Enhancement – Our concept proposes to start with this existing unified-mathematical infrastructure and enhance it to build-out a diversified collection of science & engineering modeling infrastructures, which will be known by the generic umbrella name "MetaScience". More precisely, we plan to add the rapid-automation (RA) scaffolding to the MetaCalculus infrastructure so that it naturally evolves and proliferates in end-user contexts, such as engineering schools, throughout the country.

This RA scaffolding consists of well-known software automation techniques, such as syntax macro processors, compiler-compilers, GUI builders, and special website generators that we have tailored and integrated into an "ensemble of evolution escalators" to organize and automate "do-it-yourself" production of new modeling and solution media. It is believed that the cost to deploy, operate, and support the evolutionary use of this complete ensemble (including MetaCalculus) on a suitable supercomputer is less than ten man-years per year. This is based on the history of development of the ensemble and the high degree of automation involved.

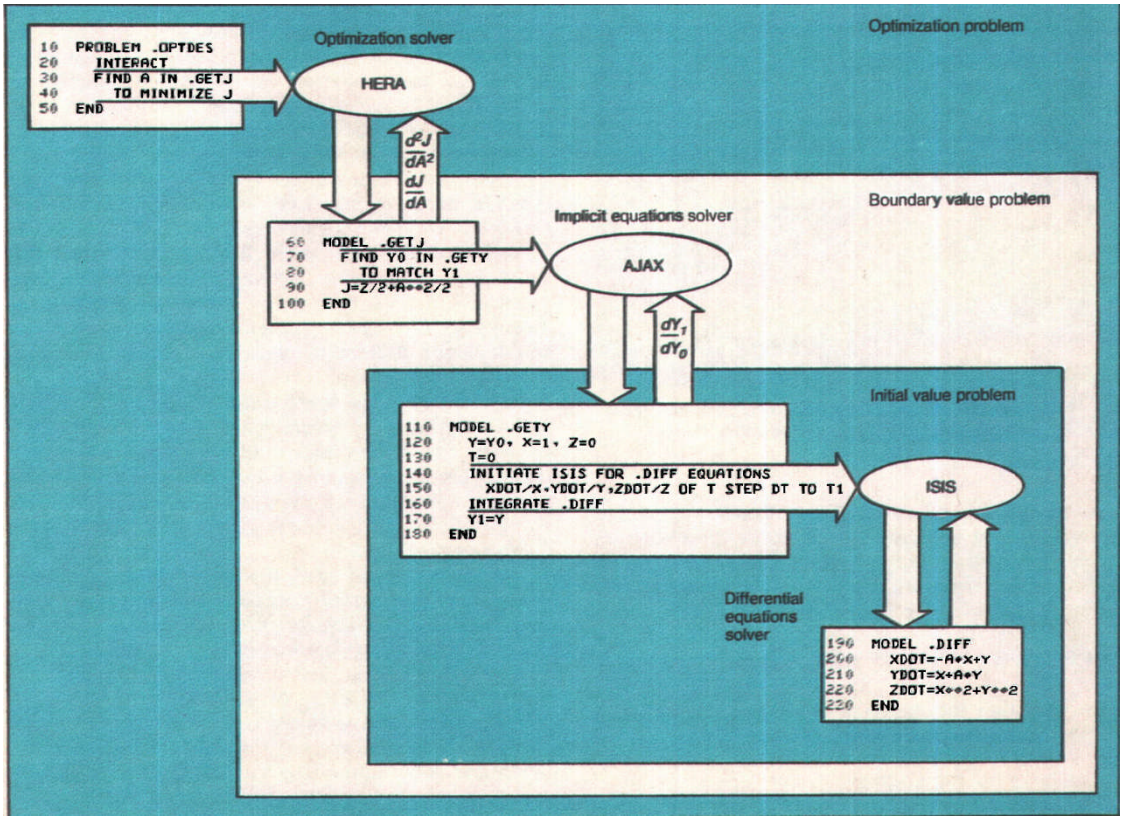


Figure 4 TSPROSE Version (6th Generation)

Catalysis of Extreme Modeling – Experience has shown that this infrastructure concept has a natural propensity for catalyzing sociological reformation from its pattern of usage. Implications are that it will reform the division of labor, such that computer science and application science will progressively settle into an organic peer relationship, producing high productivity leverage from "agile" methods overcoming the inefficiencies of the traditional Waterfall development method. The reformation mechanism is ***optimization re-engineering*** of legacy simulation code in MetaCalculus languages, following a software recycling trend similar to the "componentization" trend of object-oriented programming (OOP), which largely enabled the growth of the open-source software movement.

This optimization re-engineering can fuel the build-out of MetaScience media, extending the canned mathematics of MetaCalculus media into vertical market (domain-specific) MetaScience media containing canned science and engineering. Another compelling reason for this is the fact that parallelized supercomputing is much easier to achieve in domain-specific design than in general-purpose design.

MetaCalculus Overview

This section provides more detail in the use of the 3-tiered MC modeling alphabet. Figure 5 illustrates the essential syntax of another application utilizing this same design pattern of a two-point boundary value problem of differential equations nested within a (constrained) optimization problem, solving for ten unknowns and satisfying two constraints. The differential equation is the second-order cantilevered-beam equation, used to represent an aircraft wing.

Figure 6 shows more detail of this application to illustrate its semantic processing. The blue arrows indicate the "calculus" calls of the language, which trigger iterative processes of differential search and integration. The first arrow on the left starts the constrained optimization search to find the two vectors

(**eis** and **alpha**) using the Thor solver, a sectionally-linearized LP algorithm. The dimension of these vectors is the number of wing sections in the cantilevered wing. The **eis** vector represents the flexural rigidity of each wing section. The **alpha** vector represents the length of each section.

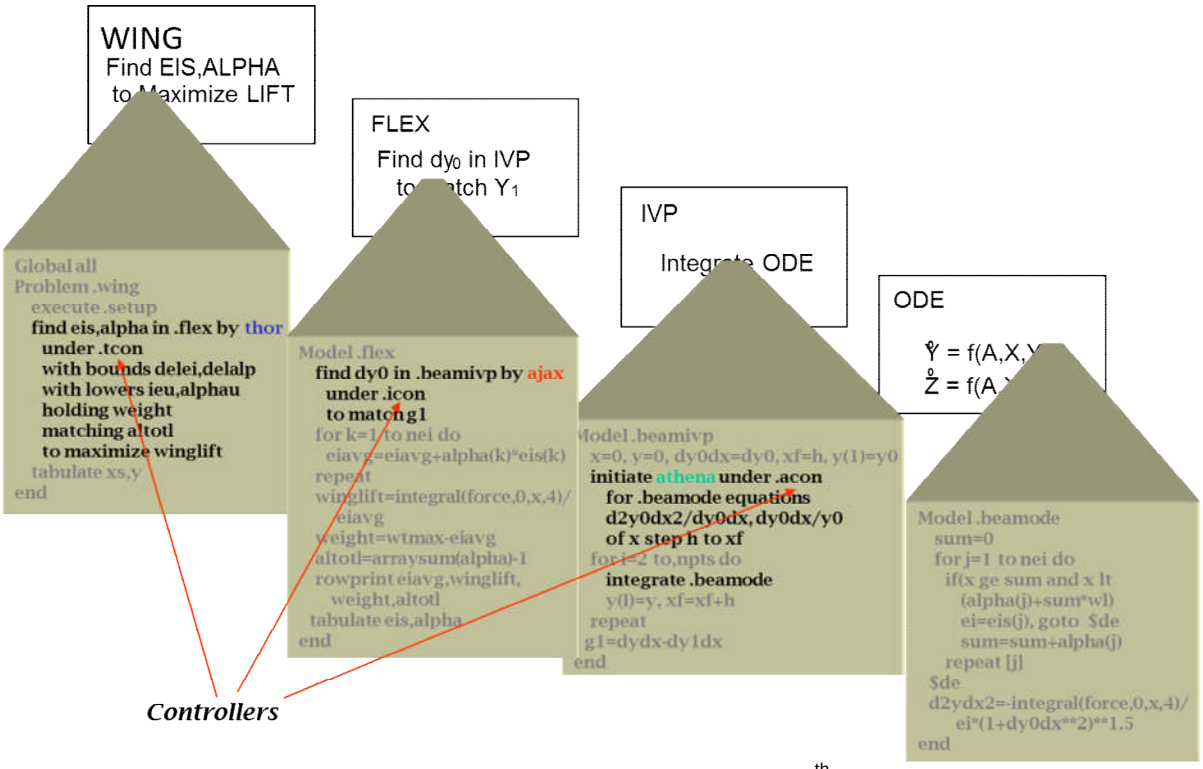


Figure 5. Wing Design Optimization in TSPROSE (6th Generation)

The **under** phase of Thor's **find** statement invokes a controller **.tcon** (not shown). This is a portal into the Thor solver to set Thor's control parameters to values different from their default values (if required). The two phrases **with bounds** and **with lowers** define vectors corresponding to the unknown vectors **eis** and **alpha** that represent step bounds on the changes in these vectors during each search iteration, and lower limits on the unknowns (constant constraints). The **holding** phrase defines the variable inequality constraint **weight**, which must be held greater than or equal to zero. The **matching** phrase defines a variable equality constraint **altotl**, which must be matched to zero within a specified tolerance (a solver control parameter).

This **find** statement calls the solver Thor, which then iteratively calls the model **.flex**, invoking algorithmic differentiation (AD) with respect to the **eis** and **alpha** vectors each time. Since the first statement of **.flex** is another **find** statement, it calls the find's solver, Ajax (A first-order Newton method). Ajax then detects that AD is active, and saves any partial derivatives that may have been accumulated. Then it restarts AD with respect to its unknown vector (the scalar dy0 in this case), and calls its model **.beamivp**. This model's sub-problem is an initial-value problem to integrate the second-order cantilevered beam differential equation, whose rate variable is d2ydx2.

The first line of the **.beamivp** model sets the initial conditions of the initial-value problem. The initial rate variable dydx is set to the dy0 variable which is the unknown of the **find** statement in the model **.flex**. The next statement, **initiate**, defines the differential equations in the integrand model **.beamode** to be integrated, and initializes the numerical-integration solver Athena (a variable-order, variable step-size Runge-Kutta method). This statement resolves the second-order differential equation into two first-order equations by defining each equation's rate variable on the left of a slash (integrator input), and its

corresponding state variable on its right (integrator output). It identifies the integration variable (x) in the **of** phrase, the suggested integration step size (h) in the **step** phrase, and the upper limit of the integration interval (xf) in the **to** phrase. Next a **for** loop is used to cycle over piecewise integration of the **.beamode** model via the **integrate** statement, increasing the limit variable xf by the interval h, each time. The reason for this piecewise integration is to collect the integrated variables in the vectors **yb** and **xb** for printing purposes. The **integrate** statement actually calls the integrand model **.beamode** multiple times for each integration step according to the needs of the solver Athena to meet a precision tolerance.

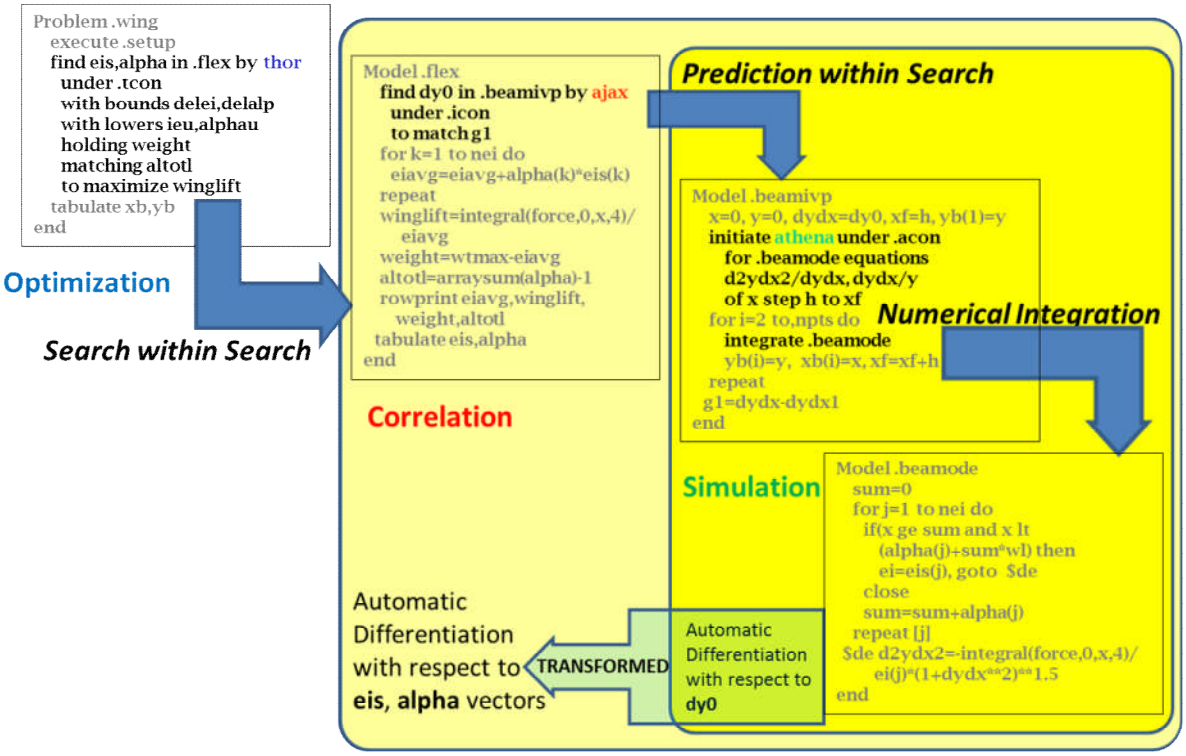


Figure 6. A profile view of the wing optimization problem

After the **repeat** statement, ending the **for** loop, the boundary value constraint variable g1 is computed as the difference between the output rate, dydx, and the specified boundary condition, dydx1. It is the zero value of this constraint that is searched for by the **find** statement (and its solver Ajax) in the **.flex** model, to solve for the initial condition dy0. Ajax is a Newton method which requires the derivative of g1 with respect to dy0 in its search process. While the determination of this derivative would be impossible via a symbolic differentiation method, it is readily determined by algorithmic differentiation of all the arithmetic occurring in and downstream of the **.beamivp** model, including the quadrature integral in the differential equation formula (the last statement in the **.beamode** model) and the integration solver Athena.

The Ajax solver invoked in the **.flex** model **find** statement executes the **.beamivp** model iteratively until the g1 constraint converges to zero. But before returning, Ajax calls a differential coordinate system transformation subroutine, which transforms all of the derivatives with respect to dy0 computed in and downstream of the **.beamivp** model into partial derivatives with respect to the **eis** and **alpha** vectors (the NAD process). Then the algorithmic differentiation continues with respect to **eis** and **alpha** for the subsequent computations of the **winglift** objective function and the two constraints **weight** and **altotl**, constituting the principal variables processed by the Thor solver in the optimization iterations.

Transcending Algorithmic Programming – The NAD semantics architecture represents automated intelligence which obviates the need for much of the procedural skill of algorithmic programming, in the same way that Fortran obviated the need for even understanding the processes of machine arithmetic. It is automated far beyond the need to understand algorithmic logic in application programming. Modelers using MC are hardly aware that by merely configuring the templates inside models, they are automatically synthesizing such powerful composite solvers under the hood, "glued" together dynamically by the differential geometry sensitivity information provided by the NAD semantics.

The cascade of modeled sub-problems is comprehended intuitively, because it mirrors the expected science behavior understood at a higher conceptual level. The implication of this distinction between MC modeling and conventional algorithmic programming is that in the future, scientific software will only be understandable by those who model the science, because the ad-hoc nested semantics will be too convoluted for *logical* comprehension. This will broaden the participation of scientists and engineers, enabling them to computerize new applications that are simply not feasible today because of the burden of numerical analysis and the delay of using languages unsuited to higher mathematics.

From the modeling perspective, it is not essential to know all of this calculus semantics in order to solve problems at this higher level. It is like the mechanics of an engine that the automobile driver is not required to understand. For this reason, we believe that modeling and solving pedagogy can eventually be pushed downward using technologies like serious games, from the level of this subject, the calculus of variations, to introductory pre-algebra in middle school. A great deal of the intervening mental gymnastics of symbolic reasoning in math pedagogy is simply not required to quantitatively understand science.

Higher Modeling & Program Generation Media

Mathematical Application Template Systems (MATS) - The potential for higher orders of modeling automation was demonstrated using PROSE as an algorithm evolution platform in 1975 by McDonough, currently University of Kentucky professor of Mechanical Engineering and Mathematics. Then a staff mathematician at PROSE, Inc., McDonough developed several PROSE test programs for validation and comparison of AD with symbolic and finite difference (approximate) differentiation. He also redeveloped the GRG algorithm [55] in PROSE, for later "subordinization" into the PROSE library as a solver engine.

EPOC – The most important example was his development of a mathematical application template system for dynamic programming modeling, called EPOC (economic planning via optimal control). This established an important new class of modeling tools—the "model-input" MATS class. Actually, this had not been the first model-input MATS. It was preceded by the *Apollo Propulsion Analysis Program* (APAP) [19,20,44-47], which added the MC1 translator to Alford's correlation engine[19] developed at TRW.

But EPOC was the first MATS developed in a **very**-high-level language, using the templates of PROSE to reach a "**very**-**very**-high-level" of modeling. It involved implementation of Pontryagin's Discrete Maximum Principle algorithm in EPOC, which did not fall prey to the "curse of dimensionality" of Bellman's dynamic programming algorithm. This MATS algorithm used PROSE's built-in Thor optimizer for stage-wise optimization. In test programs, McDonough ran nonlinear optimal control applications having 10,000 decision variables on a memory-limited CDC 6600 in 1975. EPOC was applied to a Timberland Investment Management Model (TIMM) for Simpson Timber Co. (Seattle) and the Western Timber Association (San Francisco) to compute the harvesting and regeneration policy for Western forests. (McDonough and Park [34,35]),

Platform for Higher Modeling Architectures – MATS program generators are middleware architectures for higher modeling. A major boost in the state of the art provided by FORTRAN II in 1958, was that it became a platform upon which to evolve more abstract modeling languages and template-architecture program generators, which triggered their growth in the 2nd computer generation era. Similarly, MetaFor, the planned Fortran 95-compatible successor to Fortran Calculus, can renew that vertical-evolution growth. It provides the modeling middleware for implementation and production of model-input-MATS program generators, which only contain canned mathematics, and canned-model-

MATS program generators, which are built upon canned engineering modeling-component libraries as well as canned math libraries. Examples of each type are briefly reviewed below.

Model-Input MATS Example – Forrester's System Dynamics motif, as currently implemented in STELLA [38], VenSim [39], and PowerSim [40], is representative of the Model-Input MATS class. We have postulated a GUI design that would combine MetaCalculus superstucture modeling with System Dynamics substructure modeling.

Figure 7 illustrates how this type of GUI would operate. Associated with the model of an MC System Dynamics Simulation (SDS) holon. It would enable the model to be inserted into MC design patterns like the ODC pattern of Figures 4, 5, and 6, using the SD motif GUI, and automatically translated into MC semantics for solution. This is illustrated in Figure 7 via the inset showing the flow and stock symbols characteristic of System Dynamics modeling motif. This extended MC interface would allow the wealth of SD modeling, produced over the last 50 years, to be nested within MetaCalculus hierarchies to address inverse problems of design optimization and scientific method correlation of models containing differential equations, masked by Forrester's pedagogy of stocks and flows.

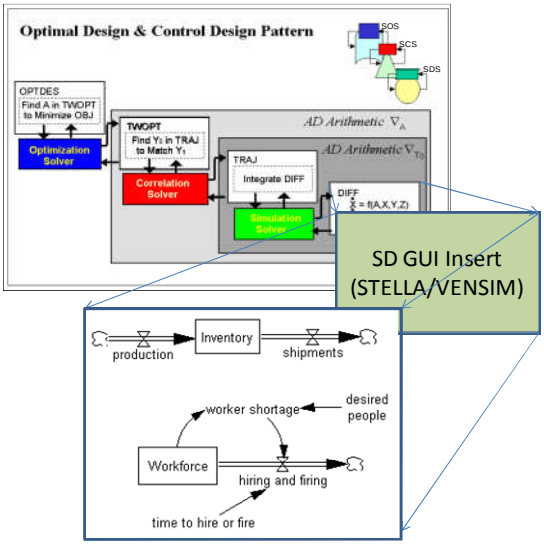


Figure 7. MC[SD] GUI Architecture

Canned-Model MATS Example – NASA's Shuttle interactive mission-planning software, the Flight Design System (FDS-2) [42], which contained a compendium of Shuttle engineering model components in its canned simulation processors is representative of the Canned-Model MATS class that could be further elevated to optimization via MetaCalculus platforms. It was designed in the late 1970s, by re-engineering Apollo batch-mainframe mission planning software into an interactive CAD system featuring *game-like* mission visualization using Tektronix 4014 graphics terminals. Figure 8 shows snapshots of such images. The FDS design was specifically "lowered" by this canned engineering approach for use by non-degreed technicians having no engineering modeling skills, and was deployed on dedicated mini-computers. It was used to design all the Shuttle missions, primarily by 8A minority set-aside contractors operating in this low skill "serious game" mode. The FDS, and other NASA software evolved from it, were used for operational planning of the International Space Station. They exemplify the re-engineering build-out of legacy software into higher media for mission modeling.

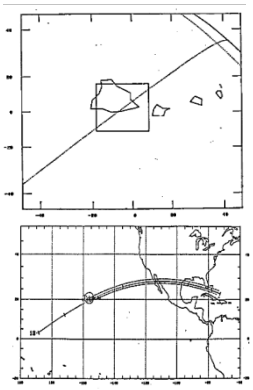


Figure 8. FDS Images

Peer Catalytic Theory of Optimization Re-Engineering

The most important leverage provided by Fortran Calculus and MetaFor is the capability for re-engineering legacy code for simplified optimization, such as all of the models embedded in the old Fortran of FDS into automatically differentiable optimization components, which can be re-synthesized into new design optimization applications.

The SLANG program in Figure 2, the TSPROSE program in Figure 4, and the Fortran Calculus program in Figure 3 were all examples of re-engineering of a Fortran optimal design and control (ODC) application [18] (Figure 1), showing 5 to 1 leverage when combining all of the MC templates. The Wing Design Optimization problem of Figures 5 and 6, has the same design pattern as the simpler ODC problem, but solves for 10 simultaneous constrained unknowns for a five section wing. The same program could just as easily have posed a 100-section wing.

In 1975 an engineer at a California manufacturing firm was using BASIC to simulate an AC motor design, when introduced to TSPROSE by a sales-support mentor from its vendor. In a few hours they had converted the BASIC to an optimization program to solve for 12-design parameters, subject to 8 constraints, which produced a 30 percent gain in motor efficiency. The elapsed time spent was a minute fraction of the time the engineer had already spent trying to solve for the parameters by trial and error simulation. The mentor who assisted him did not have any understanding of the engineer's electric-motor equations, but didn't need to. The mentor's role was to apply the MC solvers to any equation model. It was the engineer's job to understand the model.

This episode became the inspiration for a theory *of* ***catalytic*** *learning pedagogy* that we believe can transform the undergraduate and graduate educational research agenda, and catalyze "extreme modeling" to enhance discovery and understanding, the way OOP did in computer science during the last two decades.

Understanding the science and engineering embedded in these old models (e.g. how to convert such science and engineering into equations in the first place) is not necessary in the re-engineering. It can be done by computer-science students, who would ordinarily not be exposed to such science. Yet such code porting can be a catalyst for automated learning of a ***vital complementary skill***. The major benefit to the computer science students will be learning how to apply the solution-engines of MetaCalculus to arbitrary equation systems whose imbedded science they do not have to understand. Yet they can recognize recurring design patterns, which can be abstracted into new MATS templates. With this skill they can become optimum collaborators, and part-time mentors, to scientists and engineers of all disciplines who do the actual modeling (converting science to equations) *and thereby diffuse* ***the burden of science diversity*** *which the computer scientists no longer have to cope with. Yet with MATS opportunity discoveries, they can advance the modeling art vertically with new development*.

The MetaScience Mandate – Thus we have a prescription for a "MetaScience Re-Engineering and Renaissance" build-out agenda. Computer Science students and professionals, currently overwhelmed by the diversity of science and engineering when they have to produce the code, can learn a complementary role *synergistic with but without the burden of all that diversity*, by simply re-engineering old simulation code for optimization; thereby becoming "optimization mechanics" able to collaborate in extreme modeling with all of the diverse science and engineering "drivers" who do the modeling by converting science to equations. In effect, the computer-scientists can become *solution engineers who are modeling catalysts, but not involved in science itself, and yet become the developers of new modeling tools.*

We believe that with a national agenda, encouraged by a grant program, and led by university engineering departments as an outgrowth of the supercomputer provisioning planned by this project, this transition to a *peer division of labor* could be accomplished (reach critical mass of growth) in less than five years. Such a national agenda would encourage contribution by industry and government sources of old simulation software source code to be used as "archeology strata" by students to extract the modeling code from and perform optimization-re-engineering of it. This kind of term project and thesis research would "osmotically" train students to become mentors in the kind of peer collaboration that occurred originally between the TSPROSE mentor and the AC motor design engineer.

Moreover, the recycled simulation modeling, now purified of the algorithmic solution code of the "coded calculus" era before the advent of algorithmic differentiation, can be collected and re-configured by OOP methodology into modeling alphabets. The synergy of application science and computer science would thereby leverage with complementary differences to computerize new models with extreme speed and find solutions with extreme computational performance. Computer science can then become "solution science" in the truest sense. This will be the broad objective of the provisioned university research enhancements of the CI:ADDO project that we will plan for in this project

Spiritext—the MetaScience Pedagogical Medium

The web has made possible a new kind of dynamic software documentation that is integrated "in situ"

with source code [31]. Here we provide a brief discussion of its context-sensitive invocation of pop-up manual windows, and its wiki-side use for collaborative notes. For semantic-intensive MetaCalculus languages, which invoke large solver engines from libraries, the bulk of the Spiritext can be generated by the parser (part of the compiler), and includes the entire content of all of the user guides and reference manuals that would ever be needed.

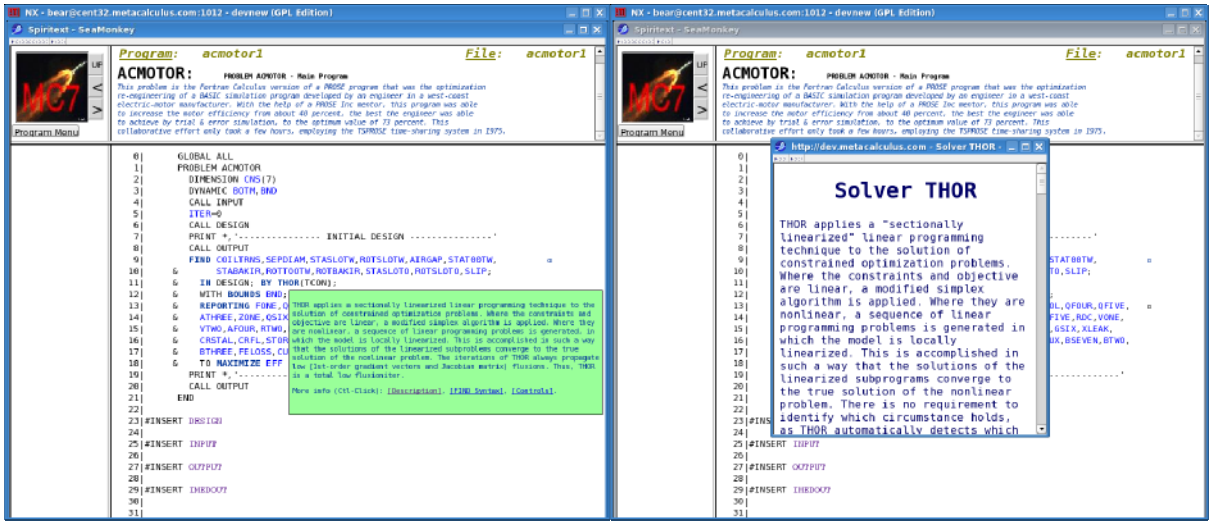


Figure 9. Screenshots showing link-click in “THOR” tooltip invoking popup description

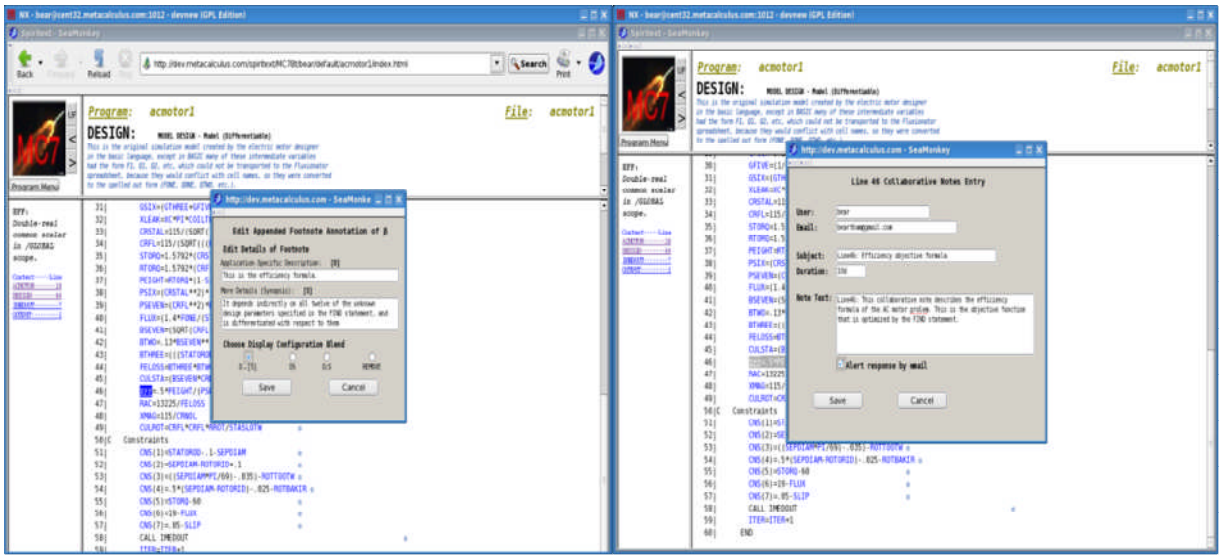


Figure 10: Adding permanent footnotes and temporary collaborative notes

Such Spiritext generation illustrated in Figure 9 shows two Spiritext screenshots generated from the AC electric motor optimization program discussed above, here written in Fortran Calculus. The upper left frame is a logo with navigation buttons. The upper right frame contains the synopsis of the source code listing in the lower right frame. Its content is taken from an external file referenced in a comment field on the first line of the source-code patch, but deleted from its presentation in the lower right frame. Although the screen capture program used to capture these images does not show the actual mouse pointer, its location (over “THOR”) is what causes the Spiritext effects seen in the images. A Control click on a link in the tooltip invokes the user-manual popup on the right.

Using wiki methods, augmented by the user via the web-browser, it can deploy many learning concepts discussed in the early 1990s by leading learning theorists [22]. In particular, it automatically provides an Extensible Web of Explanation where "people demonstrate their understanding by revising and extending their explanations" [25]. It utilizes multiple-linked representations [29] to automatically integrate a "Meta-Level" of Instruction [30] in which the pedagogical content itself is the "mini-manual" used to context-sensitively index the "maxi-manual", constituting all relevant web-accessible documentation, including, in the case of MC programs, a collection of *different* application examples having similar design patterns, such as Figures 1-6. Another class of permanent tooltip footnotes and temporary "collaborative notes" are added by the user (reader) of the Spiritext web-pages, using dialog boxes, as shown in Figure 10. The collaborative note on the right sends an email message with a return link to the position of its footnote in the code, inviting the email recipient to view the note and comment through the same medium. These notes have temporary time-limits set in the dialog box. We believe this sub-hypertext will come into its own in the use of tablet computers as MetaScience cloud portals.

Rapid Automation (RA) Scaffolding

The goal of this initiative is to re-institute an end-user movement toward automation of extreme modeling via a strategy like that of the original development of MetaCalculus at TRW—the CUE (Computer Utility for Engineers) Project [50] which evolved the five generations establishing the MetaCalculus paradigm of PROSE in five years during the Apollo program. RA evolution of modeling languages using tools like the ML/I syntax macro processor [51] was the legacy arising out of 2nd generation computing, subsequently lost in the staple software recycling trend that prevailed with the IBM 360. Staple (mass application) functionality, subsequently infused with graphic arts, morphed into an interface-style programming agenda with the advent of GUIs, and then client-server web interfaces.

In all of this style focus, much commercial PC software development by the year 2000 had settled into the recycled functionality of circa-1980 mainframe software, albeit dressed up in ever-changing styles, and becoming ever more labor intensive in development. Two human generations had gone by since the end-user modeling quest of the 1960s, which had produced MetaCalculus—dedicated toward one-of-a-kind computerization in time to meet the narrow opportunity windows (NOW) of R&D. Today initiatives toward "agile programming" are taking over software engineering to try to overcome the inefficiencies of the Waterfall specification-driven process. But as a recent Dr. Dobb's linked article [56] warned, these initiatives will not scale without automation.

End-user software engineering is getting much attention these days with recent publications, notably [57]. But coming largely from computer-science research, rather than end-user application-science origins, they hover around interface technology, most notably spreadsheets, and have not reached down into middleware yet. Consequently they do not focus on critical core automation technologies, like syntax macro processors, compiler-compilers and GUI builders that are needed to build-out end-user media.

Having long ago achieved the kind of empowerment for those end-users having the most dire needs (STEM professionals in education and R&D), which are barely mentioned in [57], we propose to establish a movement that renews the Apollo build-out of MetaCalculus (canned mathematics) media, toward the automated modeling of MetaScience (canned science & engineering) media, automated by the leveraging technologies mentioned, but restored to their original RA vertical-evolution purpose by special scaffolding which renews their leverage, discussed below.

Metacybernetics – Technology for Leveraging Vertical Evolution – After the evolution of PROSE and TSPROSE, we embarked upon the development of "blank slate" computer architecture at the Aerospace Corporation [9], intended to become a platform for hardware-assisted MetaCalculus, following the interpretive virtual-machine ("metacomputer") mode of MC2-MC6. Metacybernetics [52] was a RA technology of generating and evolving such languages and metacomputers from "highest-order metaphoric" designs, which could potentially be evolved by end-user application developers, in the same kind of context and short timeframes in which MetaCalculus had evolved.

Later with the development of MC7 (Fortran Calculus) on Intel 386, Vax, and Cray computers, we were able to avoid the slower metacomputer mode of MetaCalculus execution on such conventional hardware by bifurcating the variables of the Fortran native-code runtime environment to achieve differential-arithmetic execution performance on the order of Fortran itself. So the need for the blank-slate hardware architecture was overtaken by high-performance semi-conductor technology.

But in considering the abstraction path to be taken in the vertical evolution from unified MetaCalculus to diversified MetaScience, involving the upward build-out of metaphoric media (languages and GUI composers), the Metacybernetic automation principles were needed again, because computer-science provisioning had once again focused on the low-level design of optimizing hardware with their popular metaphoric tools—compiler-compilers like Yacc [63], and GUI builders like Glade [64] which feeds the Gtk+ API libraries [65]. Both had degenerated into metaphoric C-preprocessors, in which the metaphoric end-user "top" part of the design (BNF grammar or widget pallet), in which changes were "mutations" of the whole, took a back seat to the "bottom" part—*implementation design* in C or other languages, in which changes were "surgical". There was no way to back-reference surgical changes into mutational changes, so the two design levels became disconnected. Vertical evolution consequently became frozen (highly labor intensive to make metaphoric design changes).

GAEMY [53] and EverGlade [54]: Escalators for Rapid Media Evolution – To compensate for this problem, we developed wrapper prototypes for Yacc (Bison [66]) and Glade, which automate the synchronization of both design levels using Metacybernetic principles and modern tools such as the Meld [67] editor and the Git [68] version control system for synchronizing multi-levels of source code design and directory trees, even when different-levels of design are being compared. The degree of automation is such that novice programmers can easily adapt and successfully evolve whole systems (synchronized metaphoric design and implementation design) in languages they are not even familiar with. Two generations of prototypes have been developed in each of these categories.

In the case of GAEMY, its wrapped base (Bison) is highly stable, so most of its improvements affect the rapid prototyping of new languages by a new method of rapid-iterative-grammar assembly. But its intended use is to translate higher language grammars into MetaCalculus grammars (***very-very****-high-level source to* ***very****-high-level source—relatively simple translation, easy for students to master*) to produce "wrapper translators" as preprocessors to MC translators. So the *rapid evolution of diversified MetaScience modeling languages can be easily accommodated even in application science domains, such as engineering departments.*

The two EverGlade prototypes, EG2 and EG3 wrap Glade2 and Glade3, which have taken far different paths. Glade2 has been used for producing well respected tools, such as the Geany IDE [58]. But Glade3 has eschewed the original approach of producing C or other language interface programming images and stubs (for downstream surgical design), and produces only XML description of the metaphoric GUI design, relying on advances of the Gtk2 API library to directly interpret XML GUI descriptions at runtime, enabling the metaphoric design of totally dynamic GUIs. With the advanced features of EverGlade, which produces GtkPerl GUI interpreters, this opens broad possibilities for producing interactive GUIs, even on supercomputers.

Compelling New Research & Education Opportunities

Dynamic Correlation and Optimization Paradigm Extensions

Alford and McDonough earlier produced two higher algorithm classes which are logical extensions of the MC paradigm, both of which employed earlier MC incarnations as "power-tool platforms".

The first of these was GOP (General Optimization Program), an optimal control trajectory optimization program designed to rapidly prototype detailed bipropellant ("engine-balance") propulsion models involving implicit differential equations. It was developed at TRW in 1963, and applied a "quasilinearization" [18] algorithm (a 2nd order Newton method applied in function space). It was for this

project that Alford developed the 2nd-order NAD semantics—automatic differentiation and differential geometry coordinate transformations.

The second was MAFIA (Mack Alford's Filtering Algorithm), which became the engine of the Apollo Propulsion Analysis Program (APAP), which used MC1 as its model compiler. MC1 became the progenitor of the subsequently evolving MC paradigm, but the full sophistication of the GOP or MAFIA algorithms were never subsequently implemented in any of the MC platforms, as the opportunities to do so never arose.

EPOC, discussed previously, was a third species of the same class.

As all three of these dynamic optimization algorithms have been implemented in MC context previously, re-implementing them as a new dynamic engine class is straightforward and can be accomplished quickly.

MOB Searching – Painless Multicore Parallel Optimization

Another straightforward extension is MC Optimization Broadcasting. The same model can be simultaneously searched by many different optimization solvers from many different starting points, and the results of all of these parallel jobs can be collected in a database, and with special RunViewer GUI tools, non-convergent jobs can be quickly diagnosed, using the detailed reports that are automatically generated by all the MC solvers. A key advantage is that this is a straightforward approach to global optimization, and can be applied just as readily to nested inverse problems involving optimization of differential-equation models as to simple parameter optimizations.

The same MOB searching can be simultaneously applied to the same model with many different objective functions. So models can evolve very quickly and be thoroughly evaluated in just a few runs, albeit covering very broad search footprints. Thus NOW timeframes can be satisfied with global optimization even for very difficult optimization applications.

MOB Dynamic Correlation and Optimization – An obvious extension of MOB searching is the broadcasting of dynamic correlation and optimization super-solvers to deal with global optimization and pathological objective functions, as well as simultaneous application of different objective functions (different identification and estimation functions in the case of correlation).

Swarm Hyper-surface Data Mining & Visualization – An extension of MOB searching is the potential automation of data-collection of all of the MOB jobs to synthesize (interpolate) whole hyper-surface domains, not just search paths. This will involve applying coordination intelligence between the MOB solver instances, and spawning of interpolation searches to march along equality constraints to produce contour curves. The intent is to rapidly characterize the hyper-surface in the database for immersive and projective visualization graphics.

PDE Solver Algorithm Research

In many ways, our interest here is to continue what we started in the 2nd computer generation with the development of SLANG and PROSE, as referenced in the survey paper by Nilsen & Karplus [59]. This paper referenced SLANG and PROSE as simulation languages, because that is what the survey was about. But the authors were careful to point out that our technology transcended the art of simulation, providing the ability to optimize continuous systems through the generation of Jacobian matrices during each iteration.

They went on to survey six simulation languages dedicated to the modeling and solution of partial differential equations, SALEM, PDEL, LEANS, DSS, PDELAN, and FORSIM. Our desire is to build PDE solvers in which mathematical reduction to ODEs is unnecessary in the code itself, but is rather accomplished under the hood from direct declarations of the PDEs themselves, like in the languages referenced above.

Recently our team had occasion to review an interesting PDE correlation application developed by Dr. Lee Zia in his PhD research [60]. It was a PDE parameter-estimation problem using a method we had been researching as a potential new MetaFor PDE solver. McDonough [62] and Scofield [61] both wrote papers on this topic, after reviewing Zia's thesis.

Growing Engineering Education

We strongly believe that the NSF should institute a program of optimization componentization of re-engineered simulation models by mining past engineering software code bases, such as NASA's Shuttle Flight Design System [42], using it to educate future scientists and engineers. Redirecting the object-oriented motive of IT *recycling* of staple business functions, toward highly diversified "archeology" of recent science and engineering is a way of killing many birds with one stone. It can motivate students to pursue STEM careers (first or second) with historical examples that will enliven their studies with real drama, such as what really caused the Challenger and Columbia accidents, or even Apollo 13. It will re-train the strata of IT OOP professionals in supporting engineering modeling while these practitioners create compendia (libraries) of modeling components in many diversified fields that can be used to synthesize new system-optimization models, and *automate* the art of modeling. Each engineering department should have one or more projects of this kind.

Identifying Community Consensus Needs

There are several important capability tradeoffs that need to be examined with regard to computing research communities, especially in view of the precedents of industry software that have been shaped by three decades of consumer-oriented commercial software, and software-engineering techniques of developing and delivering staple (mass-application) software for others to use over life cycles; as opposed to one-of-a-kind software developed by end-user science and engineering researchers, where the end-item is not software, per se, but knowledge gained from research. Programs in this case are simply temporary end-items that are in constant evolution as experimental apparatus.

"High gradient" evolution is especially important at the outset of R&D programming development, when the problem to be solved (the "embryo model") is only partially understood. What is called for is rapid learning by prototyping and experimentation to arrive at a valid model and program design, like for example: what design parameters should be solved for, what the most important objective function is, what the constraints are, and how they should be formulated. Seldom can these questions be resolved without experimentation, and it must be quick so as to be cheap in manpower cost.

Questions of functionality versus cosmetics take on totally different import in this context. While GUIs may be very attractive for the staple software user, they are notoriously difficult to develop, whereas command-line menus providing the very same functionality can be quickly produced and adapted by end-users, without significantly diverting them from the primary research tasks.

The steps to be employed to assess community consensus needs are discussed in the project plan below.

Project Plan of the CI-P Project

As the intent of this initiative is to broadly introduce and recruit a team from the university community to participate in a MetaScience infrastructure build-out to begin in the Spring of 2013, following submission of a CI:ADDO proposal in October 2012, the project plan of this CI-P proposal will be to provision this introduction and recruiting. Because of our affiliation and long association with professor McDonough at the University of Kentucky, and his close association with the Center for Computation Sciences and the DLX Cluster supercomputer there, we propose this center and this supercomputer to be the hub and pilot center for the CI:ADDO project. Should the status of this center and supercomputer change during the interim, we would propose to adapt the planning accordingly to accommodate the new circumstances.

To familiarize the local UK community and other university communities with the MetaCalculus technology and its scaffolding in order to broaden participation in the planning, we propose to install the current MC7 infrastructure on the DLX supercomputer, or its surrogate, and begin training the community in its capabilities. As this installation (or its equivalent—making the existing cloud installation available for access) can be accomplished immediately upon funding the CI-P project, McDonough is preparing to teach a 500-level course in MetaCalculus at the Current CCS media room in McVey Hall during the Spring semester, 2012. Using the media production facilities there, this course will be recorded as a webinar, to be used in community recruiting for the CI:ADDO proposal effort to begin in late Summer of 2012.

Initial Infrastructure Installation

The initial installation will include the currently available MC7 Fortran Calculus language, translator, kernel and solver library plus the Spiritext scanner (whose usage is shown in Figures 9 and 10), manuals, web-portal and test-bed scaffolding tools. The MC7 translator combines with the resident Fortran compiler to implement programs written in Fortran Calculus, an extended modeling dialect of Fortran 77. The MC7 kernel bifurcates floating point variable cells to serve dual roles as real number values in non-differentiation contexts, and either real-number constants of differentiation or integer pointers to partial-derivative arrays in differentiation contexts. It performs nested algorithmic differentiation as invoked by the templates of the language.

Contributed MC7 Vendor Nexus Product Installation

MetaCalculus, LLC, our for-profit affiliate will contribute its proprietary MC7 nexus product, install and maintain it in closed-source binary form on the supercomputer. This nexus includes the Fortran Calculus translator, and the runtime kernel and library (KERLIB). Installation will be performed and tested by Joseph Thames, PI, and president of both the for-profit (Vendor) and non-profit companies.

The MC7 Library contains 18 operational optimization search engines and three in development. It contains three correlation engines and two in development. It contains ten differential equations system integrators, and seven quadrature solvers. It contains a graphics engine which interfaces the DISLIN graphics library from the Max Planck Institute.

Contributed MSF Scaffolding Tools

In addition to the Vendor supplied nexus product, the MSF open-source scaffolding ensemble, to be used for vertical evolution of modeling enhancements, will be installed, and training webinars produced to train users in their application and evolution. The MSF ensemble includes:

- MIDUS – Menu Interfaced Development and Usage System – the primary executive used to control all of the MSF ensemble; an integrated development environment (IDE) implemented as command-line menus, which can be interfaced via Linux desktop terminal emulators or via SSH;
- Spiritext Scanner Generator - This is the Spiritext website generator for Fortran Calculus;
- EverGlades – Two RA GUI application escalators, wrappers for the Glade2 and Glade3 GUI builders, designed for end-user GUI development icon and pallet-based modeling composers;
- GAEMY – Grammar Action Evolution Manager for Yacc, wraps the Bison (Yacc) parser generator, enabling the rapid evolution of application languages and translators for extending MetaCalculus languages to domain-specific MetaScience modeling languages;

All of these tools are maintained by the Git version-control system which operates as common substructure, maintains infrastructure repositories, and is controlled by simple menus. Taken together these tools are the RA means to produce highly diversified MetaScience modeling technologies built upon MetaCalculus languages in the future CI:ADDO build-out. Their installation as part of the CI-P project will be for training the initial community personnel, including students, faculty, and staff, and identifying build-out subprojects to be proposed in the CI:ADDO proposal, next fall.

Steps to Assess Community Consensus Needs

Following the initial training and exposure to the community, a website will be prepared as a statistics gathering questionnaire to assess community needs. This will be in the nature of a manifesto explaining the intent of the proposed CI:ADDO project build-out to follow, and eliciting comments from the community expressing their needs as proposed users of the infrastructure. We will also invite them to qualify themselves based on their education and research experience, more or less in the manner in which publishers qualify their demographics when offering free trade publications. These qualifications will be used to weight the sampled results. Major tradeoff questions regarding needs-priorities will be posed, such as those of expert researchers vs. novice/trainees, "built-for" vs. "do-it-yourself", permanent products vs. ad-hoc exploration apparatus, development vs. research, etc.

This website will become a community outreach vehicle and will invite ideas expressing needs and requirements for new domain-specific MetaScience appliances to be produced in the CI:ADDO project. It will encourage the development of affinity groups to advocate and pursue new modeling appliances.

Inasmuch as the overall intent of this initiative is to shift the focus and focal point of computerization from manual programming (converting equations to code) to the higher purpose of automated modeling (converting science to solutions), it is hard to imagine that there is a limit to the useful lifetime of the contemplated infrastructure.

As to the assessment of costs to create/enhance and operate the infrastructure, it is already known that this is most affected by the respective roles of manpower vs. automation and the division of labor, which are manifestly sociological issues. We have seen examples of 100 to 1 gains in productivity and speed of development as the result of shifting from Waterfall to end-user development because the Waterfall effectively subordinates the priority of highly diversified and volatile technology, engineering modeling, to the priority of stable technology, numerical analysis, causing iterated labor in long feedback loops to multiply the cost of changing models (i.e. evolution). It is this kind of analysis that will most characterize estimation of costs in this project.

Expected CI:ADDO Enhancements

Inasmuch as we are seeking funding from private as well as public sources, the actual work we will propose for the CI:ADDO proposal may have significantly shifted by next Fall. However certain planned enhancements are envisioned at this juncture that will be likely included in the CI:ADDO proposal. These enhancements fall into 3 general classes based on the organizations involved:

1. Vendor Proprietary Enhancements – Those enhancements to the closed source MC Kernel and Library and product suite performed by vendor personnel;
2. Collaborative Enhancements – Those enhancements in which the vendor and MSF assume an architectural overview and technical direction role, but which are largely performed by university personnel, including staff, postdocs, faculty, graduate and undergraduate students using open-source tools available in the marketplace and additional scaffolding tools provided by the MSF, including the prototype tool families EverGlade and GAEMY, and Spiritext Generators.
3. University Research Enhancements – Those substantially new algorithms and models initiated and developed by staff, faculty, and students of the universities involved with the infrastructure.

Vendor Enhancements

The primary vendor "supercomputer" enhancements consist primarily of internal parallelization of existing library algorithms and external parallelization of optimization-job ensembles, known as "MetaCalculus Optimization Broadcasting" (MOB) searching, discussed above. The current MC library contains 15 constrained optimization solvers which will be internally modified for two levels of parallelization and external job-assembly apparatus will be produced to aggregate MOB job collections to run as a unit on a cluster supercomputer or distributed to independent cloud servers.

Collaborative Enhancements

A major intent of the collaborative enhancements is to transfer vendor technology to the university community to cause it spread via the cloud to all colleges, including those under-represented in STEM development. It is meant to foster an automation-engineering agenda that can blossom and proliferate throughout all engineering and computer-science schools. The purpose is to automate modeling by advancing to higher-order media (languages and GUIs) which funnel through the MetaCalculus computing nexus on supercomputers. The intent is unify modeling and programming so that is it done at a much higher level interface at which *science is converted to equations*, rather than the current relatively low-level computer-science interface in which *equations are converted to code*.

The technology to be transferred involves essentially four categories:

1. MetaFor 95 Translation – The cannibalization of the GNU G95 compiler to produce the MetaFor language translator and spiritext publisher, intended to subsume and supersede conventional Fortran;
2. Interactive Optimization Supercomputing Interfaces – Implementation of web and tablet desktop-GUI interfaces for interactive optimization of MOB jobs on a Cluster supercomputer;
3. Extensibility Scaffolding – Implementation and workshop-utilization webinar development for the use of ML/I, GAEMY, and EverGlade scaffolding for the production of advanced modeling interfaces, including language extensions, new modeling languages, composer GUIs for MC programs, and modeling GUIs like the MC[SD] GUI reviewed above;
4. GUI Production – Use of the Extensibility Scaffolding to produce previously designed composer GUIs and Modeling GUIs, as pilot projects for instituting the proliferation of this discipline in the various engineering schools in the U.S.

University Research Enhancements

The permanent or “steady-state” purpose of this technology transfer effort will be to provision the continuing development of higher-order modeling appliances that can achieve rapid prototyping of new-concept techniques and products in the short time-frames that students have to accomplish significant research, and industry has to prove new concepts for commercial benefits. Historically this has depended upon platforms which eased the vertical evolution of software from the complexity of the base hardware logic progressively nearer to the highest levels of conceptualization by humans.

Use of Proceeds

The funding of this CI-P project will be entirely used to support the PI, Joseph Thames and senior team members Mack Alford and Frank Germano, including travel, lodging, and office space at the UK in Lexington, KY for the approximately seven months up to and including the submission of the CI:ADDO proposal next fall. Since they are also the developers of the software ensemble to be installed on the UK supercomputer, this is a synergistic use of their talents and skills.

Their availability at the UK will enable them to work with Professor McDonough and selected CCS staff to perform the installations, prepare documentation, and conduct briefings as required to move the agenda along and recruit participants in the initiative. The schedule of activities of these three, plus other senior team members, Mickish and Scofield, will be determined in early planning meetings of this CI-P project.

References

[1] *PROSE – A General Purpose Higher Level Language*, *Procedure Manual*, Control Data Corp. Pub No. 840003000 Rev. B (Jan 1977). [Available at www.metacalculus.com/prosemanuals.html]
[2] *PROSE – A general Purpose Higher Level Language, Calculus Operations Manual*, Control Data Corp. Pub. No 840003200 Rev B (Jan. 1977). [Available at www.metacalculus.com/prosemanuals.html]
[3] *PROSE – A general Purpose Higher Level Language, Calculus Applications Guide*, Control Data Corp. Pub No. 84000170 Rev. A (Jan 1977). [Available at www.metacalculus.com/prosemanuals.html]
[4] *PROSE – A general Purpose Higher Level Language, Time Sharing System Guide*, Control Data Corp. Pub. No 84000160 Rev A (Jan. 1977). [Available at www.metacalculus.com/prosemanuals.html]
[5] J.M. Thames, Computing in calculus, Research/Development, (1975), pp. 24–30 [Available at http://www.metacalculus.com/doc/PROSE/Computing_in_Calculus.pdf]
[6] F. W. Pfeiffer, Automatic differentiation in PROSE, ACM SIGNUM Newsletter, 22 (1987), pp. 1–8 http://www.metacalculus.com/doc/PROSE/AD_in_Prose.pdf
[7] J.M. Thames, The Evolution of Synthetic Calculus: A Mathematical Technology for AdvancedArchitecture, in *Proc. of the International Workshop on High-Level Language Computer Architecture*, University of Maryland, 1982 [Available at www.metacalculus.com/wisc.html]
[8] B. Krinsky and J. Thames, The Structure of Synthetic Calculus, A Programming Paradigm ofMathematical Design, in *Proc. of the International Workshop on High Level Computer Architecture*, University of Maryland, 1984 [Available at www.metacalculus.com/wisc.html]
[9] A.E. Speckhard, T.C. Wood, and J.M. Thames, The Aerospace Research Computer, a Status Report, in *Proc. of the International Workshop on High-Level Language Computer Architecture*, University of Maryland, 1982 [Available at www.metacalculus.com/wisc.html]
[10] Edsger W. Dijkstra (1982). "On the role of scientific thought". In Dijkstra, Edsger W.. *Selected writings on Computing: A Personal Perspective*. New York, NY, USA: Springer-Verlag New York, Inc. pp. 60–66. ISBN 0-387-90652-5
[11] J.W. Forester, System Dynamics and Learner-Centered Learning in Kindergarten through 12th Grade Education (http://sysdyn.clexchange.org/sdep/Roadmaps/RM1/D-4337.pdf)
[12] Debra Lyneis, "The Future of System Dynamics and Learner Centered Learning in K-12 Education, Part II", A Report from the Planning Meeting in Essex, MA, June 23-July 1, 2001, http://www.clexchange.org/ftp/newsletter/CLEx12.2.pdf
[13] Debra Lyneis and Lees N. Stuntz, "System Dynamics in K-12 Education: Lessons Learned", The Creative Learning Exchange, 2007 http://www.systemdynamics.org/conferences/2007/proceed/papers/LYNEI390.pdf
[14] J.M. Thames, *FORTRAN Calculus: A new Implementation of Synthetic Calculus*, Digital Calculus Corp., Torrance, CA (1989) [Available at www.metacalculus.com/fc77.html]
[15] *Fortran Calculus User Manual*, Digital Calculus Corporation (1990) [Available at www.metacalculus.com/FCManChapters.html]
[16] J.M. Thames, Synthetic Calculus – A Paradigm of Mathematical Program Synthesis, in [17]. Preprint version: http://www.metacalculus.com/doc/FC/Synthetic_Calculus.pdf
[17] A. Griewank and G.F. Corliss, eds., Automatic Differentiation of Algorithms: Theory, Implementations, and Applications, SIAM, Philadelphia (1991)
[18] R.E. Bellman and R.E. Kalaba, *Quasilinearization and Nonlinear Boundary-Value Problems*, American Elsevier Publishing Company, New York (1965).
[19] M.W. Alford, Mathematical Aspects of the Flight Analysis Computer Programs, AIAA Paper 68-583, *Fourth Propulsion Joint Specialist Conference*, Cleveland, Ohio, June 10–14, 1968
[20] J.C. Hooper, Performance Analysis of the Ascent Propulsion Subsystem of the Apollo Spacecraft, NASA Program Apollo Working Paper MSC-03408 [Available at www.metacalculus.com/apollo.html]
[21] M.W. Alford, A Computationally Stable Noise-in-the-State Filtering Algorithm, AIAA Paper 68-887, presented at the *AIAA Guidance, Control, and Flight Dynamics Conference*, Pasadena, CA, August 12–14, 1968
[22] David N. Perkins, Judah L. Schwartz, Mary Maxwell West, and Martha Stone Wiske, Editors, *Software Goes to School—Teaching for Understanding with New Technologies,* Oxford University Press, New York, 1995.
[23] David N. Perkins, David Crismond, Rebecca Simmons, and Chris Unger, "Inside Understanding"

(Chapter 5 in [22]).
[24] Judah L. Schwartz, “Shuttling Between the Particular and the General: Reflections on the Role of Conjecture and Hypothesis in the Generation of Knowledge in Science and Mathematics” (Chapter 6 in [22]).
[25] Judah L. Schwartz, “The Right Size Byte: Reflections of an Educational Software Designer” (Chapter 10 in [22]).
[26] Arthur Koestler (1968). *The Ghost in the Machine*. McMillan: New York.
[27] Herbert Simon (1968). *The Sciences of the Artificial.* MIT Press: Boston.
[28] Ken Wilber (2000), A Brief History of Everything, Revised Edition, Shambala, Boston
[29] E. Paul Goldenberg, “Multiple Representations: A Vehicle for Understanding Understanding” (Chapter 9 in [22])
[30] Stephen H. Schwartz and David N. Perkins, “Teaching the Metacurriculum: A New Approach to Enhancing Subject-Matter Learning” (Chapter 14 in [22]).
[31] Joseph M. Thames and Stephen W. Duckett, “Computer-Based User Interface for A Memory-Resident Rapid Comprehension Document for Original Source Information”, United States Patent Application Publication US 2004/0189713 A1, Sep 30, 2004.
http://www.freepatentsonline.com/20040189713.pdf
[32] Emile Durkheim, *The Division of Labor in Society*, 1893, The MacMillan Co., 1933, Free Press edition,, 1964, p. 183.
[33] Judah L. Schwartz, “Can Technology Help Us Make the Mathematics Curriculum Intellectually Stimulating and Socially Responsible?”, International Journal of Computers for Mathematical Learning 4: 99-119, 1999, Kluwer Academic Publishers.
[34] J. McDonough and D. Park, A Discrete Maximum Principle Solution to an Optimal Control Formulation of Timberland Management Problems, Presented at *Western Forest Economics Conference*, Wemme, OR, May 1975. [Available at www.metacalculus.com/epoc.html]
[35] J. McDonough and D. Park, Nonlinear Optimal Control Approach to Interregional Management of Timber Production and Distribution, *Proceedings, Systems Analysis Workshop, Society of American Foresters*, University of Georgia, 1975
[36] Thomas Kuhn, *The Structure of Scientific Revolutions*, University of Chicago Press, 1962,
[37] George Corliss, et al, editors, *Automatic Differentiation of Algorithms—From Simulation to Optimization*, Selected papers from the Third International Conference on Automatic (Algorithmic) Differentiation, Cote d’Azur, France, (June 2000), Springer, 2002.
[38] ISEE Systems, “STELLA—Systems Thinking for Education and Research”,
http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx
[39] Ventana Systems, Inc. “Vensim Software—Linking systems thinking to powerful dynamic models”,
http://www.vensim.com/software.html
[40] Powersim Software, “Powersim Studio 8 Products” ,
http://www.powersim.com/main/products___services/powersim_products/
[41] Jim Hines, “Modeling with Molecules 2.02”, Ventana Systems, Inc. 2005,
http://www.vensim.com/molecule.html
[42] J.M. Thames and W.M. Chunn, “An Evolving Expert System for Shuttle Experiments Flight Planning”, Paper presented at USAF Space Division Industry Symposium, Vandenberg, AFB, 1985.
http://www.metacalculus.com/doc/FDS/Evolving_FDS_Expert_System.pdf
[43] J M. Thames, “Flight Analysis of the Apollo Propulsion Systems”, NASA Program Apollo Working Paper No. 1196, NASA Manned Spacecraft Center, March 8, 1966 [Available at
http://www.metacalculus.com/doc/Apollo/Flight_Analysis_of_the_Apollo_Propulsion_Systems.pdf]
[44] R.K.M. Seto, “Apollo 9 Mission Report: Descent Propulsion System Final Flight Evaluation”, MSC-PA-R-69-2, Supplement 8, NASA Manned Spacecraft Center, August 1970 [Available at
http://www.metacalculus.com/doc/Apollo/Apollo_9_Descent_Propulsion_System_Flight_Analysis.pdf]
[45] R.K.M. Seto, “Apollo 10 Mission Report: Descent Propulsion System Final Flight Evaluation”, MSC-00126, Supplement 7, NASA Manned Spacecraft Center, December 1969 [Available at
http://www.metacalculus.com/doc/Apollo/Apollo_10_Descent_Propulsion_System_Flight_Analysis.pdf]
[46] R.J. Smith and W.G. Griffin, “Apollo 12 Mission Report: Service Propulsion System Final Flight

Evaluation", MSC-O1855, Supplement 3, NASA Manned Spacecraft Center, December 1971 [Available at http://www.metacalculus.com/doc/Apollo/Apollo_12_SPS_Flight_Evaluation.pdf]
[47] A.T.Avvenire and S.C. Wood, "Apollo 15 Mission Report: Descent Propulsion System Final Flight Evalutation", MSC-05161, Supplement 4, NASA Manned Spacecraft Center, September 1972 [Available at http://www.metacalculus.com/doc/Apollo/Apollo_15_DPS_Flight_Evaluation.pdf]
[48] M. Campbell-Kelly and D.D. Garcia-Swartz, "Economic Perspectives on the History of the Computer Time-Sharing Industry, 1965-1985, IEEE Annals of the History of Computing 1058-6180/08. January-March, 2008.
[49] J.M.Thames, "SLANG—A Problem-Solving Language of Continuous Model Simulation and Optimization", ACM National Conference, San Francisco, 1969. [Available at http://www.metacalculus.com/doc/SLANG.pdf]
[50] J.D. McCully, "The Q Approach to Problem Solving", Proceedings of the Fall Joint Computer Conference, 1969.
[51] P. J. Brown, "The ML/I Macro Processor", Communications of the ACM (October 1967). [Current Website: http://www.ml1.org.uk/]

[52] J.M. Thames, "Metacybernetics – Design Evolution of Metacomputer Host Architecture", The AeroSpace Corporation, 1983, [Available at http://www.metacalculus.com/doc/WISC/Metacybernetics.pdf]

[53] J.M. Thames, "GAEMY – Grammar-Action Evolution Manager for Yacc" [Available at http://www.metacalculus.com/Metacybernetics/GAEMY.html]

[54] J.M. Thames, "EverGlade – Glade GUI Evolution Manager" [Available at http://www.metacalculus.com/Metacybernetics/EG.html]

[55] J. Abadie and J. Carpentier, "Generalization of the Wolfe Reduced Gradient Method to the case of Nonlinear Constraints," in Optimization, R. Fletcher (Ed), 1969.

[56] Electric Cloud, "Five Reasons why Agile Won't Scale Without Automation",

[57] Ko, A., Abraham, R., Beckwith, L., Blackwell, B., Burnett, B., Erwig, M., Scaffidi, C., Lawrence, J., Lieberman, H., Myers, B., Rosson, M., Rothermel, G., Shaw, M., and Wiedenbeck, S. 2011, "The State of the Art in End-User Software Engineering". ACM Computing Surveys.

[58] Geany IDE [http://en.wikipedia.org/wiki/Geany]

[59] R.N. Nilsen and W.J. Karplus, Continuous System Simulation Languages: A State of the Art Survey, Annales de l' Association Internationale pour le Calcul analogique, Paris, January 1974 [Available at http://www.metacalculus.com/doc/PROSE/Continuous_System_Simulation_Languages.pdf]

[60] L.L. Zia, Applied Mathematics and Computation, 30, 19-47 (1989);

[61] D.F. Scofield, Constrained Optimization Approach to the Problem of Fitting 2D Diffusion Models to Empirical Data Sets, (private communication), Available at http://www.metacalculus.com/doc/PDE_Solvers/Scofield-2.pdf]

[62] J.M. McDonough, Parameter Estimation for a Convective-Diffusive PDE using MetaFor (private communication) Available at [http://www.metacalculus.com/doc/PDE_Solvers/McDonough-1.pdf]

[63] S.C. Johnson, Yacc—Yet Another Compiler Compiler, [http://dinosaur.compilertools.net/yacc/]

[64] The Glade Interface Designer [http://glade.gnome.org/]

[65] The Gtk+ Project [http://www.gtk.org/]

[66] GNU Bison [http://en.wikipedia.org/wiki/GNU_bison]

[67] Meld Diff & Merge Tool [http://meld.sourceforge.net/]

[68] Linus Torvalds, Git—the Fast Version Control System [http://git-scm.com/]